UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/479,363	01/07/2000	Timothy James Graser	RO999-122	2954

24038 7590 02/17/2004

MARTIN & ASSOCIATES, LLC
P O BOX 548
CARTHAGE, MO 64836-0548

EXAMINER	
LY, ANH	

ART UNIT	PAPER NUMBER
2172	11

DATE MAILED: 02/17/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

BR4

Office Action Summary	Application No.	Applicant(s)	
	09/479,363	GRASER, TIMOTHY JAMES	
	Examiner	Art Unit	
	Anh Ly	2172	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 December 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-23* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Response to Arguments

1. Applicant's arguments filed on 06/16/2003 with respect to claims 1-19 have been considered but are moot in view of the new ground(s) of rejection.

2. Claims 20-23 have been added.

3. Claims 1-19 are pending in this application.

Claim Rejections - 35 USC § 103

4. 2.The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 3-5 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,405,209 issued to Obendorf in view of US Patent No. 6,317,748 issued to Menzies et al. (hereinafter Menzies).

Regarding to claim 1, Obendorf teaches an apparatus for instantiating and initializing an object from a relational database. As shown in FIG. 1 is an exemplary hardware that has ***at least one processor; a memory coupled to the at least one processor.*** **As shown in FIG. 3B is a reference table as *class configuration data***

comprising a plurality of entries residing in the memory, each class configuration entry including a key-value pair, wherein the key includes **TableName object ID as** ***information relating*** **to the process of creating the table object as** ***a selected processing context and the value includes*** **the class I D as** ***configuration data for a class in the selected processing context.***

Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed ***an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data.*** and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B). Obendorf does not explicitly teach wherein the key comprises context information appended to a class identifier.

However, Menzies discloses attaching key properties to a qualifier key to each property that constructs the key for the class (col. 9, lines 38-45).

Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to combine the teachings of Obendorf with the teachings of Menzies so as to obtain the attached to the key component of the key for the class of object (col. 9, lines 7-62). This combination would modify the apparatus for being included an object oriented class replacement mechanism in order to instantiate and initialize an object. And the Management Information Base (MIB) objects are mapped to a singleton class by accessing a mapping table to translate the MIB object corresponds to a table collection such as a keyed managed object format object (Menzies – col. 2, lines 40-67).

Regarding to claim 3, Obendorf teaches all the claim subject matters as discussed in claim 2, and further discloses *the class identifier comprises a class token that comprises a text string* (Fig. 3B).

Regarding to claim 4, Obendorf teaches all the claim subject matters as discussed in claim 1, and further discloses *a factory object that generates an instance of the selected class by accessing the appropriate entry in the class configuration data using the key* (Col. 5, lines 20-37).

Regarding to claim 5, Obendorf teaches all the claim subject matters as discussed in claim 1, and further discloses the step of *generating tile key from a class identifier and from the context information* (Fig. 3B).

Regarding to claim 20, Obendorf teaches an apparatus for instantiating and initializing an object from a relational database. As shown in FIG. 1 is an exemplary hardware that has ***at least one processor; a memory coupled to the at least one***

processor. **As shown in FIG. 3B is a reference table as** ***class configuration data comprising a plurality of entries residing in the memory, each class configuration entry including a key-value pair, wherein the key includes*** **TableName object ID as** ***information relating*** **to the process of creating the table object as** ***a selected processing context and the value includes*** **the class I D as** ***configuration data for a class in the selected processing context.***

Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed ***an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data.*** and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B). Obendorf does not explicitly teach wherein the key comprises context information appended to a class identifier.

However, Menzies discloses attaching key properties to a qualifier key to each property that constructs the key for the class (col. 9, lines 38-45).

Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to combine the teachings of Obendorf with the teachings of Menzies so as to obtain the attached to the key component of the key for the class of object (col. 9, lines 7-62). This combination would modify the apparatus for being included an object oriented class replacement mechanism in order to instantiate and initialize an object. And the Management Information Base (MIB) objects are mapped to a singleton class by accessing a mapping table to translate the MIB object corresponds to a table collection such as a keyed managed object format object (Menzies – col. 2, lines 40-67).

6. Claims 6-8, 10-11, 13-15, 17-19 and 21-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,405,209 issued to Obendorf in view of US Patent No. 6,317,748 issued to Menzies et al. (hereinafter Menzies) and further in view of US Patent No. 6,430,564 issued to Judge et al. (hereinafter Judge).

Regarding to claim 6, Obendorf teaches a method for instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests

object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). **As seen, a ClassID 218 as *configuration data is retrieved* to pass to the creation call CoCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as the step of *instantiating the instance of the class using the retrieved configuration data.***

Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed ***an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data.*** and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B). Obendorf does

not explicitly teach wherein the key comprises context information appended to a class identifier. Menzies discloses attaching key properties to a qualifier key to each property that constructs the key for the class (col. 9, lines 38-45). In combination, Obendorf and Menzies do not teach the step of ***retrieving configuration data corresponding to the class in a selected processing context using a corresponding key that includes information relating to the selected processing context,*** although as shown in FIG. 3B is a reference table that contains the class ID as *configuration data corresponding to the class* for creating the table object as *the selected processing context* and TableName object ID as *a corresponding key that includes information relating to the selected processing context.*

However, Judge teaches a data manager manages global data within a Java Virtual Machine. The data manager maintains a data class list that stores data class identifiers associated with each data class object (Judge, Abstract). As shown in Judge Fig. 2, the list that contains class identifiers associated data class object may be maintained as a single list implemented in a single hash table, keyed either by data class name or by instance label or object ID (Judge, Col. 4, lines 20-32). Thus, if using object ID as a key for the Obendorf reference table, the corresponding class ID will be retrieved by a conventional hash technique to have the step of *retrieving configuration data corresponding to the class in a selected processing context using a corresponding key that includes information relating to the selected processing context.*

Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to combine the teachings of Obendorf in view of Menzies

with the teachings of Judge so as to obtain the attached to the key component of the key for the class of object (col. 9, lines 7-62). This combination would modify the method by using a single hash table for storing class ID and object ID, using object ID as a key to retrieve class ID in order to search a class ID in the reference table for instantiating and initializing an object and for being included an object oriented class replacement mechanism in order to instantiate and initialize an object. And the Management Information Base (MIB) objects are mapped to a singleton class by accessing a mapping table to translate the MIB object corresponds to a table collection such as a keyed managed object format object (Menzies – col. 2, lines 40-67).

Regarding to claim 7, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 6, Obendorf further discloses the step of *storing the configuration data with the corresponding key* (Obendorf, Fig. 3B).

Regarding to claim 8, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 7, Obendorf further discloses the step of *generating a key from a class identifier and from the context information* (Obendorf, Fig. 3B).

Regarding to claim 10, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 9, Obendorf further discloses *the class identifier comprises a class token that comprises a text string* (Obendorf, Fig. 3B).

Regarding to claim 11, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 6, Obendorf further discloses the step of

generating the key form a class identifier and from the context information (Obendorf, Fig. 3B).

Regarding to claim 13, Obendorf teaches a method for instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). As seen, a ClassID 218 as *configuration data* is retrieved to pass to the creation call CoCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as *an object oriented class replacement mechanism that generates an instance of a selected class.* As shown in FIG. 1 is an exemplary hardware that has *signal bearing media bearing the object oriented class replacement mechanism* (Obendorf, Col. 3, line 66-Col. 5, line 4).

Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference

table (Col. 5, lines 20-37). In other words, the technique as discussed above performed ***an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data.*** and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B). Obendorf does not explicitly teach wherein the key comprises context information appended to a class identifier. Menzies discloses attaching key properties to a qualifier key to each property that constructs the key for the class (col. 9, lines 38-45). In combination, Obendorf and Menzies do not teach the step of **using a key that includes information relating to a selected processing context to access an appropriated entry in class configuration data stored external to the class**, although as shown in FIG. 3B is a reference table that contains the class ID as *configuration data corresponding to the class* for creating the table object as *the selected processing context* and TableName object ID as *a corresponding key that includes information relating to the selected processing context.*

However, Judge teaches a data manager manages global data within a Java Virtual Machine. The data manager maintains a data class list that stores data class identifiers associated with each data class object (Judge, Abstract). As shown in Judge Fig. 2, the list that contains class identifiers associated data class object may be maintained as a single list implemented in a single hash table, keyed either by data class name or by instance label or object ID (Judge, Col. 4, lines 20-32). Thus, if using

object ID as a key for the Obendorf reference table, the corresponding class ID will be retrieved by a conventional hash technique to have the step of *using a key that includes information relating to a selected processing context to access an appropriated entry in class configuration data stored external to the class* for generating an instance.

Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to combine the teachings of Obendorf in view of Menzies with the teachings of Judge so as to obtain the attached to the key component of the key for the class of object (col. 9, lines 7-62). This combination would modify the method by using a single hash table for storing class ID and object ID, using object ID as a key to retrieve class ID in order to search a class ID in the reference table for instantiating and initializing an object and for being included an object oriented class replacement mechanism in order to instantiate and initialize an object. And the Management Information Base (MIB) objects are mapped to a singleton class by accessing a mapping table to translate the MIB object corresponds to a table collection such as a keyed managed object format object (Menzies – col. 2, lines 40-67).

Regarding to claim 14, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 13, Obendorf further discloses *signal bearing media comprises recordable media* (Obendorf, Col. 3, line 66-Col. 4, line 14).

Regarding to claim 15, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 13, Obendorf further discloses *signal bearing media comprises transmission media* (Obendorf, Col. 3, line 66-Col. 4, line 14).

Regarding to claim 17, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 16, Obendorf further discloses *the class identifier comprises a class token that comprises a text string* (Obendorf, Fig. 3B).

Regarding to claim 18, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 13, Judge teaches a data manager manages global data within a Java Virtual Machine. The data manager maintains a data class list that stores data class identifiers associated with each data class object (Judge, Abstract). As shown in Judge Fig. 2, the list that contains class identifiers associated data class object may be maintained as a single list implemented in a single hash table, keyed either by data class name or by instance label or object ID (Judge, Col. 4, lines 20-32). Thus, if using object ID as a key for the Obendorf reference table, the corresponding class ID will be retrieved by a conventional hash technique to have the step of *generating an instance of the selected class by accessing the appropriate entry in the class configuration data using the ,fey.* Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the Obendorf method by using a single hash table for ;storing class ID and object ID, using object ID as a key to retrieve class ID in order to search a class ID in the reference table for instantiating and initializing an object.

With respect to claim 19, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 13, Obendorf further discloses the step of *generating the key from a class identifier and form the context information* (Obendorf, Fig. 3B).

With respect to claim 21, Obendorf teaches a method for instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). As seen, a ClassID 218 as *configuration data* is retrieved to pass to the creation call CoCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as *an object oriented class replacement mechanism that generates an instance of a selected class.* As shown in FIG. 1 is an exemplary hardware that has *signal bearing media bearing the object oriented class replacement mechanism* (Obendorf, Col. 3, line 66-Col. 5, line 4).

Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed ***an object oriented! class replacement mechanism residing in the memory and***

executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data. and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B). Obendorf does not explicitly teach wherein the key comprises context information appended to a class identifier. Menzies discloses attaching key properties to a qualifier key to each property that constructs the key for the class (col. 9, lines 38-45). In combination, Obendorf and Menzies do not teach the step of **key from the text string class identifier and from context information**, although as shown in FIG. 3B is a reference table that contains the class ID as *configuration data corresponding to the class* for creating the table object as *the selected processing context* and TableName object ID as *a corresponding key that includes information relating to the selected processing context.*

However, Judge teaches a data manager manages global data within a Java Virtual Machine. The data manager maintains a data class list that stores data class identifiers associated with each data class object (Judge, Abstract). As shown in Judge Fig. 2, the list that contains class identifiers associated data class object may be maintained as a single list implemented in a single hash table, keyed either by data class name or by instance label or object ID (Judge, Col. 4, lines 20-32). Thus, if using object ID as a key for the Obendorf reference table, the corresponding class ID will be retrieved by a conventional hash technique to have the step of *using a key that includes information relating to a selected processing context to access an appropriated entry in class configuration data stored external to the class* for generating an instance.

Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to combine the teachings of Obendorf in view of Menzies with the teachings of Judge so as to obtain the attached to the key component of the key for the class of object (col. 9, lines 7-62). This combination would modify the method by using a single hash table for storing class ID and object ID, using object ID as a key to retrieve class ID in order to search a class ID in the reference table for instantiating and initializing an object and for being included an object oriented class replacement mechanism in order to instantiate and initialize an object. And the Management Information Base (MIB) objects are mapped to a singleton class by accessing a mapping table to translate the MIB object corresponds to a table collection such as a keyed managed object format object (Menzies – col. 2, lines 40-67).

Regarding to claim 22, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 13, Obendorf further discloses *signal bearing media comprises recordable media* (Obendorf, Col. 3, line 66-Col. 4, line 14).

Regarding to claim 23, Obendorf in view of Menzies and Judge teaches all the claim subject matters as discussed in claim 13, Obendorf further discloses *signal bearing media comprises transmission media* (Obendorf, Col. 3, line 66-Col. 4, line 14).

5. Claim 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,405,209 issued to Obendorf in view of US Patent No. 6,317,748 issued to Menzies et al. (hereinafter Menzies) and further in view of US Patent NO. 6,430,564 issued to Judge et al. (hereinafter Judge) and US Patent No. 6,438,559 issued to White.

Regarding to claim 12, Obendorf teaches a method for instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance() as the step of *initiating the creation of an instance of the replacement class.* CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). As seen, a ClassID 218 as *configuration data is retrieved* to pass to the creation call CoiCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as the step of *creating an, instance of the class according to the retrieved configuration data for the class.* As shown in FIG. 3B is a reference table that contains the TableName object ID as *generating a key that includes information relating to the current processing context.*

Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID

218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed ***an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data.*** and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B). Obendorf does not explicitly teach generating a key that comprises information relating to a current processing context appended to a class identifier for the existing class. Menzies discloses attaching key properties to a qualifier key to each property that constructs the key for the class (col. 9, lines 38-45). Obendorf and Menzies do not teach the step of *storing configuration data for the existing class using a corresponding key that includes information relating to a selected processing context; replacing the configuration data for the existing class with configuration data for the replacement class while maintaining the same corresponding key; using the generated key for retrieving the configuration data for the replacement class.* Judge teaches a data manager manages global data within a Java Virtual Machine. The data manager maintains a data class list that stores data class identifiers associated with each data class object (Judge, Abstract). As shown in Judge Fig. 2, the list that contains class identifiers associated data class

object may be maintained as a single list implemented in a single hash table, keyed either by data class name or by instance label or object ID (Judge, Col. 4, lines 20-32). Thus, if using object ID as a key for the Obendorf reference table, the corresponding class ID will be retrieved by a conventional hash technique to have the step of *using the generated key for retrieving the configuration data for the class* to generate an instance. In combination, Obendorf, Menzies and Judge do not teach the streaming objects in the distributed system.

However, White teaches a method of streaming objects for distributed system. A class ID (ACI) is provided as a protocol for converting between a java object, each class is represented by a numeric identifier. A table of the class identifiers is kept at the beginning of each serialization. A simple transformation is applied to achieve portability, so that any AGI serialization can be converted to a portable serialization, a Class Descriptor serialization (ACD). The ACD is identical to ACI except that the class identifier table beginning ACI is replaced by a table of class descriptors (White, Abstract). This technique indicates the step of storing configuration data for the existing class using a corresponding key that includes information relating to a selected processing context; replacing the configuration data for the existing class with configuration data for the replacement class while maintaining the same corresponding key.

Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to combine the teachings of Obendorf in view of Menzies and Judge with the teachings of White so as to include the step of storing and replacing

the configuration data as taught by White and the attached to the key component of the key for the class of object (col. 9, lines 7-62). This combination would modify the method by using a single hash table for storing class ID and object ID, using object ID as a key to retrieve class ID in order to search a class ID in the reference table for instantiating and initializing an object and for being included an object oriented class replacement mechanism in order to instantiate and initialize an object. And the Management Information Base (MIB) objects are mapped to a singleton class by accessing a mapping table to translate the MIB object corresponds to a table collection such as a keyed managed object format object (Menzies – col. 2, lines 40-67).

Contact Information

7. Any inquiry concerning this communication should be directed to Ann Ly whose telephone number is (703) 306-4527 via E-Mail: anh.ly@uspto.gov. The examiner can be reached on Monday - Friday from 8:00 AM to 4:00 PM.

If attempts to reach the examiner are unsuccessful, see the examiner's supervisor, John Breene, can be reached on (703) 305-9790.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to: (703) 872-9306 (Central Official Fax Number)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington, VA, Fourth Floor (receptionist). Inquiries of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3900.

AL
FEB.12th, 2004

ALFORD KINDRED
PRIMARY EXAMINER